Міністерство освіти і науки України
Сумський державний університет

МЕТОДИЧНІ ВКАЗІВКИ

для виконання лабороторних робіт на тему
«Технологія програмування на мові Java. Робота з масивами в Java»
із дисциплін
«Програмування мобільних комп'ютерних систем»,
«Прикладне програмування в телекомунікаційних системах»,
«Програмування мобільних пристроїв телекомунікацій»
для студентів спеціальностей
6.171 "Електроніка",
6.172 "Телекомунікації та радіотехніка"
денної форми навчання

Суми
Видавництво СумДУ
2017

Методичні вказівки для виконання лабораторних робіт на тему «Технологія програмування на мові Java. Робота з масивами в Java» із дисциплін «Програмування мобільних комп'ютерних систем», «Прикладне програмування в телекомунікаційних системах», «Програмування мобільних пристроїв телекомунікацій» / Укладачі О.Є. Горячев, С.М. Маценко – Суми: Вид-во СумДУ, 2017. – 33 с.

Кафедра електроніки і комп'ютерної техніки

1 Мета роботи

Метою роботи є набуття навичок програмування з використанням операторів управління і масивів в мові програмування Java.

2 Склад робочого місця.

IBM-сумісний персональний комп'ютер (ПК).

Програмне забезпечення: операційна система Windows, Java 2 SDK версії 6.0 і вище, пакет NetBeans IDE 6.1 і вище.

3 Короткі теоретичні відомості

3.1. Технологія програмування на мові Java

3.1.1. Робота з документацією

Відкрийте в папці **docs** файл **index.html**, що містить короткий огляд JDK SE 6, а також посилання на документацію по Java.

У цьому файлі виберіть гіперпосилання **Java Platform API Specification** на специфікацію інтерфейсу прикладного програмування мови Java.

В результаті переходу за гіперпосиланням відкриється Web-сторінка **C: \ Program Files \ Java \ jdk1.6.0_07 \ docs \ api \ index.html**,

що містить опис всіх пакетів, класів, властивостей (полів) і методів мови Java.

Щоб швидше отримати доступ до цієї Web-сторінці рекомендується додати її в обрані сторінки (в Internet Explorer це виконується за допомогою команди Додати в папку "Вибране" меню **Вибране**).

У лівому верхньому фреймі Web-сторінки виводиться пункт **All Classes** і найменування всіх пакетів мови Java. У разі вибору опції **All Classes** (він вибирається за замовчуванням при завантаженні Web-сторінки), то в лівому нижньому фреймі виводяться найменування всіх класів і інтерфейсів всіх пакетів Java (найменування класів виводяться звичайним шрифтом, а інтерфейсів - курсивом). Якщо лівому верхньому фреймі вибрати один з пакетів, в лівому нижньому фреймі виводяться класи і інтерфейси тільки цього пакета.

Для отримання документації по компоненту в лівому нижньому фреймі необхідно вибрати ім'я цього компонента. Після цього в правому фреймі виводиться Web-сторінка з описом компонента, що містить:

- ім'я компонента;
- ієрархію його класів-предків;
- реалізовані в компоненті інтерфейси;
- список прямих нащадків даного компонента (якщо вони є);
- короткий опис компонента;
- список полів компонента (Field Summary);
- список полів, успадкованих від компонентів-предків;
- список конструкторів компонента (Constructor Summary);

• список методів компонента (Method Summary);
• список методів, успадкованих від компонентів-предків;
• більш докладні описи полів, конструкторів і методів компонента.

Щоб отримати більш докладний опис поля, конструктора або методу, треба в списку клацнути по його імені мишкою.

3.1.2. Технологія програмування додатків Java з використанням NetBeans IDE

Пакет NetBeans IDE (надалі просто NetBeans) є потужним IDE для розробки всіх видів додатків, які підтримує мову Java, включаючи не тільки стандартні програми Java (Java SE), але і додатки рівня підприємства (Java EE) і рівня портативних пристроїв (Java ME). Адреса Web-сайту проекту NetBeans:

http://www.netbeans.org/index.html.

Пакет NetBeans містить повний набір компонент: редактор, середа компіляції і виконання, а також відладчик. Пакет працює не з програмами, а з проектами.

Проект - це група файлів програми і байт-кодів, а також установки, за допомогою яких виконується складання, виконання та налагодження цих файлів. Все програмування в NetBeans, навіть якщо програма складається з одного файлу, виконується в рамках проекту. Якщо програма містить велику кількість коду, її рекомендується розбивати на кілька файлів (зазвичай в кожному файлі розміщують один клас, хоча це і не обов'язково). Всі файли і папки проекту, що розміщуються в папці з ім'ям **ім'я-проекту**, створюються і змінюються автоматично.

Вікно NetBeans має стандартний вид (дивись рис. 1.1): зверху панель заголовка, меню і панель інструментів, знизу рядок стану, в середині вікно програми.

Вікно додатка, в свою чергу складається з трьох основних вікон:
• вікно проектів **Projects** (Проекти);
• вікно редактора
• вікно виведення **Output** (Вивід).

У вікні проектів виводяться проекти, їх компоненти - класи, а також компоненти класів: поля, конструктори і методи. За допомогою контекстного меню додавати нові класи в проект, а також перейменовувати і видаляти існуючі класи, а також виконувати деякі інші операції.

Вікно редактора може містити кілька вкладок. Відкриття нової вкладки можна виконати або за допомогою команди **Open File** (Відкрити файл) меню **File** (Файл), або подвійним клацанням миші по імені класу (наприклад, Rectangle.java) у вікні проектів. Подвійний клік мишею по імені властивості, конструктора або методу підсвічує перший рядок з визначенням властивості, конструктора або методу. Клацання мишею по знаку "×" в найменуванні вкладки закриває вкладку. Символ "*" після імені вкладки означає, що текст вкладки модифіковано після останнього збереження вмісту.

Результати виконання програми виводяться у вікні виводу. Кнопка 

вмикає / вимикає висновок на екран вікна виведення.

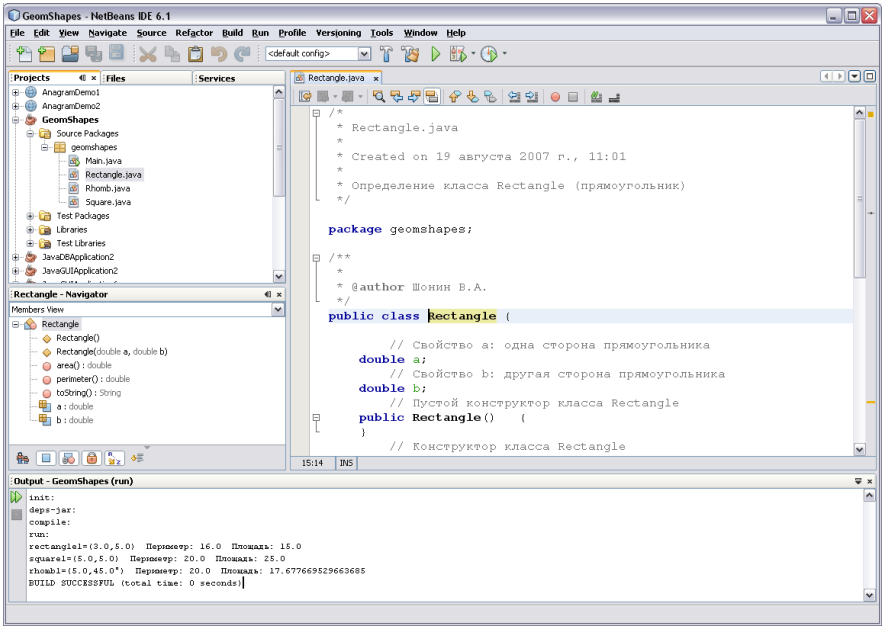

Рис. 1 - Вікно NetBeans

Створення проекту

Для створення проекту програми Java необхідно виконати наступні дії:

1. У меню **File** (Файл) виберіть команду **New Project** (Створити проект) або натисніть кнопку  на панелі інструментів.

2. У вікні **New Project** (Створення проекту) (рис. 2) виберіть в полі **Properties** (Категорії) папку **General** (Java), а в поле **Projects** (Проекти) пункт **Java Application** (Додаток Java) і натисніть кнопку **Next** (Далі).

3. У вікні **New Java Application** (Створити додаток Java) (рис. 3) в поле **Project Name** (Ім'я проекту) задайте ім'я проекту, в поле **Project Location** задайте ім'я папки, в якій буде знаходитися проект (використовуйте ім'я своєї папки на комп'ютері - обмеження: бажано, щоб шлях до папки не містив букв кирилиці).

4. Увімкніть перемикач **Set as Main Project** (Зробити головним проектом) (якщо він не включений).

5. Увімкніть перемикач **Set Main Class** (Створити головний клас) (якщо він не включений) і задайте ім'я головного класу - того класу, який містить метод **main()** (можна залишити ім'я-проекта.Main), а потім натисніть кнопку **Finish** (Завершити).

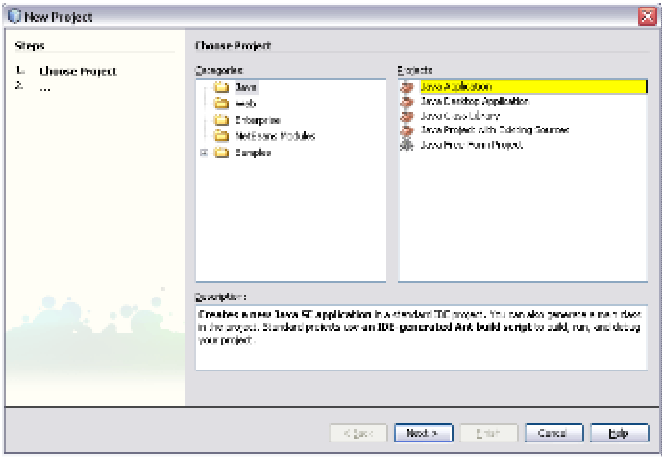

Рис. 2 - Вікно New Project

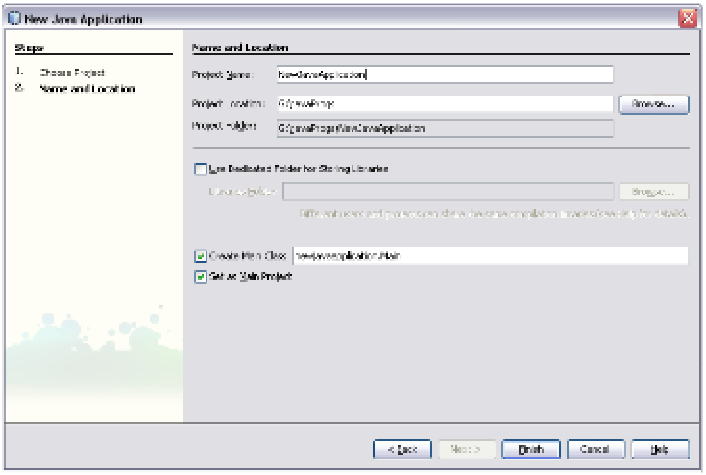

Рис. 3 - Вікно New Java Application

Операції над проектом

Існуючий проект відкривається за допомогою команди **Open Project** меню **File**, або при натисканні кнопки на панелі інструментів.

При створенні проекту для нього створюються службові файли, які становлять невід'ємну частину проекту. Тому для операцій переміщення, копіювання, перейменування і видалення проекту необхідно користуватися не засобами операційної системи, а командами **Move Project**, **Copy Project**, **Rename Project** і **Delete Project** контекстного меню проекту.

Додавання файлу в проект

Якщо проект містить більше одного файлу, то додавання файлу в цей проект потрібно виконати наступні дії:

1. Виконати команду **New File** в меню **File** або натиснути кнопку на панелі інструментів.

2. У поле **Categories** вікна **New File** (рис. 4) вибрати категорію файлу, наприклад, для нового класу, папку **Java Classes**.

3. У поле **File Types** вибрати тип файлу, наприклад, **Java Class** і натиснути кнопку **Next**.

4. Після цього виводиться друге вікно запиту на створення нового файлу. Компоненти цього вікна залежать від категорії і типу файлу. Так для нового класу вікно запиту **New Java Class** має вигляд, представлений на рис. 5. У цьому вікні досить задати ім'я нового класу (поле **Class Name**) - всі інші поля заповнюються автоматично. Після натискання кнопки **Finish** у вікні редактора виводиться шаблон для відповідного виду файлу (крім файлів типу **Java Empty Class**).

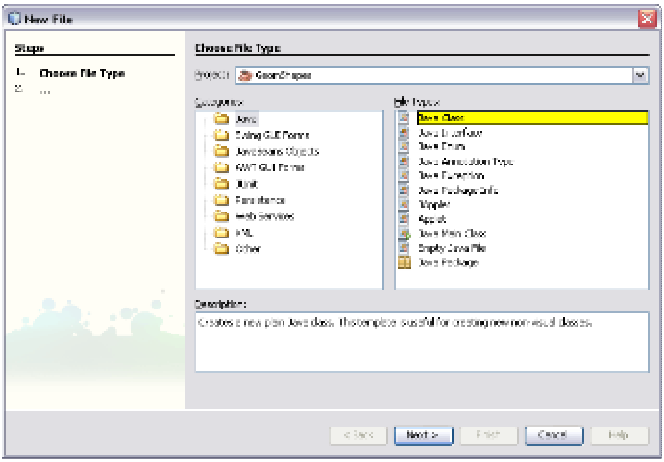

Рис. 4 - Вікно New File

Новий клас можна також створити в такий спосіб:

1. Викликати у вікні **Projects** контекстне меню для імені проекту в папці **Source Packages**.

2. У контекстному меню вибрати команду **New**.

3. У меню команди New вибрати команду **Java Class**.

4. У вікні **New Java Class** (рис. 5) задати ім'я нового класу.

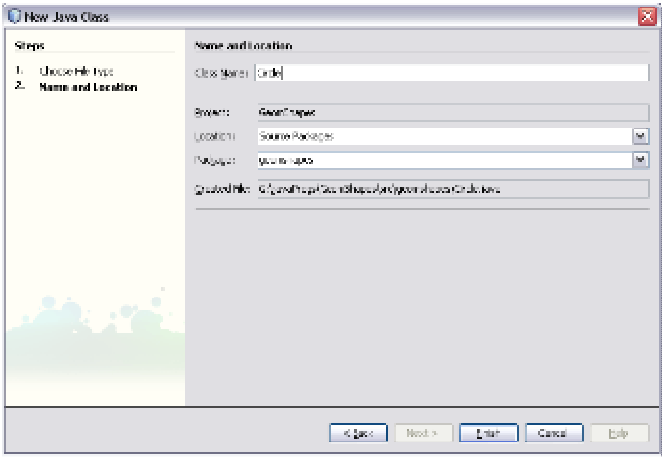

Рис. 5 - Вікно New Java Class

Операції над файлами

Існуючий файл відкривається за допомогою команди **Open File** меню **File**, або допомогою команди **Open** контекстного меню файлу.

Операції переміщення, копіювання, вставки, перейменування і видалення файлу виконуються за допомогою команд **Move**, **Copy**, **Paste**, **Rename** і **Delete** контекстного меню файлу.

Робота в вікні редактора

Інструментальні засоби редагування NetBeans включають в себе:

• набір інструментів редагування;

• панель редагування;

• засіб виведення номерів рядків вихідного файлу.

Набір інструментів редагування виводиться на панелі інструментів за допомогою включення команди **Edit** в підміню команди **Toolbars** меню **View**.

У набір входять наступні інструменти:

• кнопка **Cut**  (Вирізати) - для перенесення виділеного вмісту в буфер обміну (можна також використовувати клавіші **Ctrl + X**);

• кнопка **Copy**  (Копіювати) - для копіювання виділеного вмісту в буфер обміну (можна також використовувати клавіші **Ctrl + C**);

• кнопка **Paste**  (Вставити) - для вставки вмісту буфера обміну в позицію курсора (можна також використовувати клавіші **Ctrl + V**);

• кнопка **Undo**  (Відмінити останню дію) - для скасування останньої операції по редагуванню тексту (можна також використовувати клавіші **Ctrl + Z**);

• кнопка **Redo** (Повторити останню скасовану дію) - для повторення скасування останньої скасованої операції по редагуванню тексту (можна також використовувати клавіші **Ctrl + Y**).

Всі ці команди є також в меню **Edit**.

Окрема панель редагування включається / вимикається за допомогою команди **Show Editor Toolbar** меню **View** і має вигляд, представлений на рис. 6.

Рис. 6 - Панель редагування вікна редактора

Висновок номерів рядків файлу вихідного тексту вмикається / вимикається за допомогою команди **Show Line Numbers** меню **View**.

Набір і редагування тексту вкладки у вікні редактора виконується так само, як і в інших текстових редакторах.

Однак оскільки редактор NetBeans є спеціалізованим текстовим редактором, він має такі додаткові можливості:

• використання шаблонів для різних типів вихідних файлів;

• завершення парного введення;

• згортка / розгортка блоків;

• використання скорочень при введенні часто зустрічаються елементів вихідного тексту;

• динамічна компіляція тексту.

Використання шаблонів

При завданні типу нового файлу одночасно задається і шаблон, який виводиться в новій вкладці редактора. У редакторі для категорії **Java Classes**, використовуваної для створення неграфічних додатків і аплетів Java можна задати наступні типи і відповідні їм шаблони для створюваних файлів:

• **Java Class** - задає шаблон звичайного класу Java з порожнім конструктором без параметрів;

• **Java Empty Class** - не ставить ніякого шаблону (можна використовувати, наприклад, для завдання файлу вихідних даних);

• **Java Interface** - задає шаблон інтерфейсу Java;

• **Java Enum** - задає шаблон класу перелічуваних змінних;

• **Java Annotation Type** - задає шаблон анотацій;

• **Java Exception** - задає шаблон класу Java для опису користувацького виключення;

• **Java Main Class** - задає шаблон головного класу Java з порожнім конструктором без параметрів і порожнім методом main ();

• **Java Package Info** - задає шаблон опису пакета Java;

• **JApplet** - задає шаблон аплету Swing з порожнім конструктором;

• **Applet** - задає шаблон аплету Java з порожнім методом init ();

• **Java Package** - задає нову порожню папку для пакета Java.

Завершення парного введення

Деякі елементи мови Java обов'язково повинні бути парними. Так, якщо в тексті вводиться символ "**(**", то йому обов'язково повинен відповідати символ "**)**". Це ж стосується символів "**{**" і "**}**", " '" і "'", а також "'''" і "'''". Тому, коли вводиться відкриває символ для однієї з цих пар, редактор автоматично вставляє закриває символ пари.

Крім цього, редактор автоматично починає новий рядок, якщо натиснути клавіші **Shift + Enter** перед завершальним символом пари.

Якщо всередині набираемой рядки, обмеженою символами ' '", натиснути клавішу Enter, то новий рядок почнеться з символу "', а в кінці попереднього рядка буде вставлений символ завершення рядка ' '" і символ конкатенації рядків" + ".

Згортка / розгортка блоків

Деякі частини початкового тексту можна згортати і розгортати.

Це можуть бути:

• коментарі на початку класу;

• блоки пропозицій імпорту;

• коментарі Javadoc;

• конструктори і методи;

• внутрішні класи.

Такі частини початкового тексту позначаються зліва спеціальними графічними символами, що відзначають початок і кінець блоку (рис. 7, а). Якщо клацнути мишею по знаку "-" в квадратику, то блок згорнеться і в квадратику з'явиться знак "+" (рис. 7, б). При натисканні мишею по знаку "+" блок знову розгорнеться.

Використання скорочень

Під час введення тексту в редакторі NetBeans можна використовувати скорочення. Заміна скорочення його повною формою (шаблоном) виконується одні з двох способів:

• проста заміна скорочення шаблоном;

• заміна з вибором з можливих варіантів шаблону для даного скорочення.

При простій заміні скорочення, наведені в табл. 1, замінюються своїми шаблонами, якщо після введення скорочення натиснути клавіші **Shift + Пробіл**.

```
23          // Конструктор класса Rectangle
24      public Rectangle(double a, double b) {
25              // Присвоение значения параметра a свойству a
26          this.a = a;
27              // Присвоение значения параметра b свойству b
28          this.b = b;
29      }
30          // Метод определения периметра прямоугольника
31      public double perimeter() {
32              // Вычисление и возврат
33              // периметра прямоугольника
34          return (this.a + this.b)*2.0;
35      }
```

а)

```
23          // Конструктор класса Rectangle
24      public Rectangle(double a, double b) {
25              // Присвоение значения параметра a свойству a
26          this.a = a;
27              // Присвоение значения параметра b свойству b
28          this.b = b;
29      }
30          // Метод определения периметра прямоугольника
31      public double perimeter() {...}
```

б)

Рис. 7. Згортка і розгортка блоків: а) конструктор Rectangle і метод perimeter розгорнуті; б) метод perimeter згорнутий

Таблиця 1.

Найбільш часто використовувані скорочення і відповідні їм шаблони

Скорочення	Шаблон	Скорочення	Шаблон
En	Enumeration	Ex	Exception
Ob	Object	Psf	public static final
Psfb	public static final boolean	Psfi	public static final int
Psfs	public static final String	St	String
ab	abstract	bo	boolean
br	break	ca	catch (
cl	class	cn	continue

df	default:	dowhile	do {
			}
			while (condition);

eq	equals	ex	extends
fa	false	fi	final

fl	float	forc	for (Iterator it = collection.iterator();
			it.hasNext();) {
			Object elem = (Object) it.next();
			}

fore	for (Object elem : iterable) { }	fori	for (int i = 0; i < SCRAMBLED_WORD_LIST.length; i++) { }

Скоро-чення	Шаблон	Скоро-чення	Шаблон
fy	finally	ie	interface

ifelse	if (condition) {
	}
	else {
	}
		im	implements

iof	instanceof	ir	import
le	length	newo	Object name = new Object(args);
pe	protected	pr	private
psf	private static final	psfb	private static final boolean
psfi	private static final int	psfs	private static final String
pst	printStackTrace();	psvm	public static void main(String[] args) { }
pu	public	re	return
serr	System.err.println("|");	sout	System.out.println("|")
st	static	sw	switch (
sy	synchronized	tds	Thread.dumpStack();

th	throws	trycatch	try {
			} catch (Exception e) {
			}

tw	throw	twn	throw new

wh	while (	whilei	while (it.hasNext()) {
			Object elem = (Object) it.next();
			}

Шаблони можна редагувати і видаляти, а також можна додавати нові шаблони. Для цього необхідно в меню **Tools** вибрати команду **Options** і у вікні цієї команди натиснути кнопку **Editor**, а потім у вікні редактора вибрати вкладку **Code Templates**.

Для редагування існуючого скорочення треба в поле **Templates** вибрати скорочення, а потім в поле **Expanded Text** змінити існуючий шаблон.

Для видалення скорочення треба в поле **Templates** вибрати скорочення, а потім натиснути кнопку **Remove**.

Для додавання скорочення треба натиснути кнопку **New**, а потім в діалоговому вікні **New Code Template Dialog** в поле **Abbreviation** ввести нове скорочення і натиснути кнопку **OK**. Після цього в поле **Expanded Text** вводиться новий шаблон.

Для можливості виведення довідок для елементів мови необхідно підключити до пакету NetBeans документацію з мови (як вказувалося вище папка docs з документацією поміщається в папку jdkномер-версії). Для цього в меню **Tools** треба вибрати команду **Options** і у вікні цієї команди натиснути кнопку **Java Platforms**, а потім у вікні **Java Platform Manager** вибрати

вкладку **Javadoc** (рис. 8) і, після натискання кнопки **Add ZIP / Folder** вибрати папку **docs** в папці

c: \ Program Files \ Java \ jdkномер-версії.

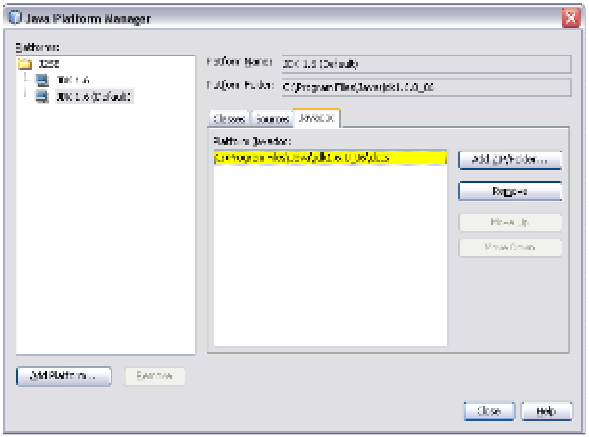

Рис. 8 - Підключення папки з документацією до NetBeans

Щоб виконати заміну з вибором, необхідно набрати початкові символи шаблону і натиснути клавіші **Ctrl + Пробіл**. Після цього над вікном редактором з'явиться вікно вибору. Якщо вибране за допомогою клацання миші пункт є об'єктом, полем або методом бібліотеки Java, для нього виводиться довідка, як це показано на рис. 9 для скорочення Str.

Видалити вікна вибору і довідки можна, натиснувши клавішу **Esc**.

Динамічна компіляція тексту

При виконанні програми в режимі командного рядка, а також у багатьох IDE, після набору і / або редагування тексту вихідний файл запускається на компіляцію. Якщо в програмі містяться синтаксичні помилки, вони передаються в вивідний потік помилок (в Java це об'єкт System.err). У редакторі NetBeans BlueJ використовується так звана динамічна компіляція, коли вихідний файл компілюється кожен раз при зміні тексту програми. Так, наприклад, в прикладі на рис. 10 в реченні в рядку 28 містяться помилки (не визначена змінна m в класі Rectangle і, крім того, в пропозиції відсутня заключний символ ";"). У цьому випадку, як видно з малюнка, зліва від пропозиції з'являється символ (якщо включена нумерація рядків, цей символ замінює номер рядка). Якщо навести курсор миші на червоний квадратик, з'являється підказка, коротко пояснює помилку. Після усунення помилок червоний квадратик зникає.

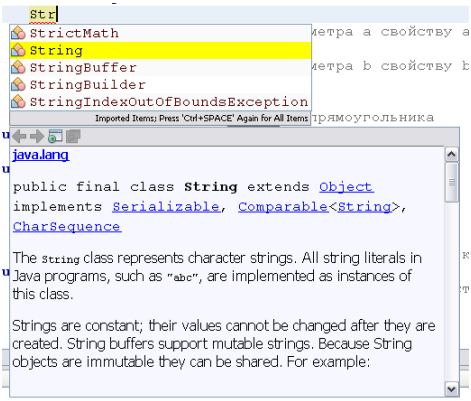

Рис. 9 - Висновок довідки для об'єкта String

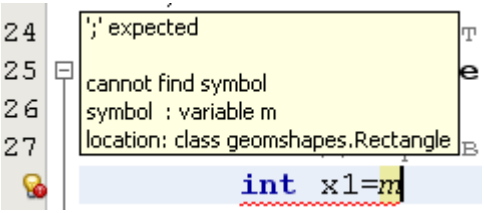

Рис. 10. Приклад синтаксичної помилки.

Інший приклад помилки, пов'язаний з перетворенням типів, наведено на рис. 11. У реченні в рядку 31 зліва від знака рівності стоїть ціла змінна типу int, а праворуч (рядок 27) - елемент строкового масиву args. Щоб виправити помилку необхідно замінити пропозицію в рядку 31 таким реченням:

```
int m = Integer.parseInt (args [0]);
```

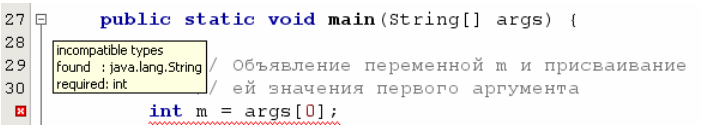

Рис. 11. Приклад помилки перетворення

Збирання проекту

Збиранням називається компіляція файлів вихідного проекту і створення файлів, що містять байт-коди класи проекту (це файли з розширенням .class). Засоби управління проектом компілюють тільки ті класи проекту, які були змінені під час останнього редагування.

Збірка проекту виконується або за допомогою команди **Build Main Project** в меню **Build**, або за допомогою команди **Build Project** в контекстному меню проекту у вікні **Projects**, або при натисканні клавіші **F11**, або при натисканні кнопки  на панелі інструментів.

Результати складання виводяться у вікні **Output**. Якщо файли проекту містять синтаксичні помилки, виводяться повідомлення про помилки. Так, для помилки, наведеної на рис. 10, висновок буде мати наступний вигляд:

```
Trying to override old definition of task java
Created dir: G: \ javaProgs \ GCDSearch \ build \ classes
Compiling 1 source file
to G: \ javaProgs \ GCDSearch \ build \ classes
G: \ javaProgs \ GCDSearch \ src \ gcdsearch \ Main.java: 36: ';' expected
            int x1 = m
1 error
BUILD FAILED (total time: 0 seconds)
```

Якщо клацнути мишею по посиланню для помилки, курсор у вікні редактора буде встановлений в рядку, що містить цю помилку.

У разі успішної збірки останньої виведеної рядком буде рядок:

```
BUILD SUCCESSFUL (total time: 0 seconds)
```

Можна також очистити проект (зазвичай ця операція виконується, якщо з проекту видаляються будь-які файли). При цьому видаляються папки, що містять файли класів проекту (папка **classes**), керуючі файли проекту (папка **build**), а також деякі допоміжні файли (папка **dist**). Частка проекту виконується за допомогою команди **Clean Project** в контекстному меню проекту у вікні **Projects**.

Збірка з чищенням виконується або за допомогою команди **Clean and Build Main Project** в меню **Build**, або за допомогою команди **Clean and Build Project** в контекстному меню проекту у вікні **Projects**, або при натисканні клавіш **Shift + F11**, або при натисканні кнопки  на панелі інструментів.

Можна виконати компіляцію окремого файлу проекту, якщо виділити цей файл у вікні Projects, а потім або виконати команду **Compile назва пакунка** в меню **Build**, або виконати команду **Compile** в контекстному меню файлу, або натиснути клавішу **F9**.

За замовчуванням, при складанні використовуються вихідні файли, розміщені в папці **src** проекту, а для тестування - файли в папці **test**. Якщо необхідно вказати додаткові вихідні файли або файли тестування, треба викликати контекстне меню для проекту і виконати команду **Properties**. У діалоговому вікні цієї команди в поле **Categories** вибрати **Source**, а потім в правій половині вікна натиснути кнопку **Add Folder** для додавання нової папки з вихідними файлами або файлами тестування.

За замовчуванням, при складанні використовуються стандартна бібліотека JDK, підключена до NetBeans. При необхідності підключення

додаткових бібліотек треба викликати контекстне меню для проекту і виконати команду **Properties**. У діалоговому вікні цієї команди в поле **Categories** вибрати **Libraries**, а потім в правій половині вікна вибрати одну з вкладок: **Compile** (бібліотеки часу компіляції), **Run** (бібліотеки часу виконання), **Compile Tests** (бібліотеки тестування часу компіляції), **Run Tests** (бібліотеки тестування часу виконання), потім натиснути одну з кнопок: **Add Project** (додавання проекту), Add Library (додавання бібліотеки) або **Add JAR / Folder** (додавання архіву або папки). Після цього вибирається необхідний проект, бібліотека, архів або папка і натискається кнопка **OK**.

Якщо при складанні необхідно вказати додаткові опції компілятора, необхідно викликати контекстне меню для проекту і виконати команду **Properties**. У діалоговому вікні цієї команди в поле **Categories** вибрати **Compile**, а потім в поле **Additional Compile Options** ввести рядок додаткових опцій.

При складанні можна також вказати, які пакети слід виключити з створюваного архівованого файлу програми Java (з розширенням .jar). Для цього треба викликати контекстне меню для проекту і виконати команду **Properties**. У діалоговому вікні цієї команди в поле **Categories** вибрати **Packaging**, а потім в поле **Exclude From JAR File** задати пакети, які треба виключити з архівованого файлу.

Вибір **Documentation** в поле **Categories** команди **Properties** контекстного меню для проекту дозволяє вказати параметри генерації документації **Javadoc** для проекту.

Виконання проекту

Виконання проекту здійснюється або за допомогою команди **Run Main Project** в меню **Run**, або за допомогою команди **Run Project** в контекстному меню проекту у вікні **Projects**, або при натисканні клавіші **F6**, або при натисканні кнопки  на панелі інструментів.

Додаткові опції для проекту перед його виконанням можна вказати, якщо викликати контекстне меню для проекту і виконати команду **Properties**, а потім в діалоговому вікні цієї команди в поле **Categories** вибрати **Run** (рис. 12).

У правій половині вікна можна задати наступні операції:

• задати інший головний клас за допомогою поля **Main Class** і кнопки **Browse**;

• задати аргументи виклику програми в поле **Arguments**;

• задати робочий каталог програми за допомогою поля **Working Directory** і кнопки **Browse**;

• задати додаткові опції віртуальної машини Java (JVM) в поле **VM Options**.

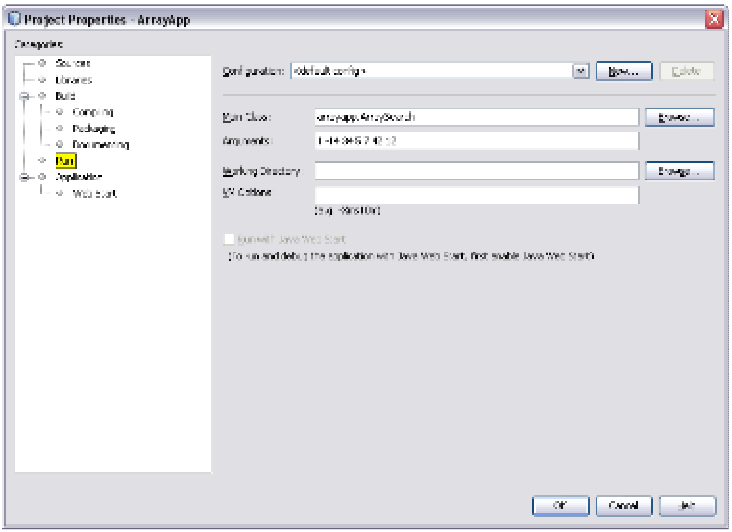

Рис. 12. Діалогове вікно категорії Run команди Properties контекстного меню пакета

Налагодження проекту

При роботі в режимі налагодження в NetBeans доступні наступні основні можливості:

• установка точок переривання;

• послідовне виконання програми;

• стеження за змінними програми.

Установка точок переривання

Перед початком сеансу налагодження можна встановити в програмі ***точки переривання***, в яких виконання програми буде припинено. У цій точці можна переглянути поточні значення змінних, почати послідовне виконання або перейти до наступної точки переривання.

В NetBeans можна визначити точки переривання для рядка, а також точки переривання для наступних елементів:

• класу (коли клас завантажується в JVM, коли клас вивантажується з JVM або коли виконується завантаження / розвантаження класу);

• виключення (коли виняток обробляється, коли виняток не обробляється або коли воно просто зустрічається в програмі);

• змінної (під час отримання доступу до змінної);

• методу (при початку виконання методу);

• обчислювального потоку (thread) (коли потік починає виконуватися, коли виконання потоку припиняється або коли потік починається / закінчується).

У редакторі рядок, в якій встановлена точка переривання, підсвічується червоним тлом, а в лівій рамці міститься значок, що позначає тип точки переривання:

- - одиночна точка переривання;
- - відключена точка переривання;
- - множинні точки переривання;
- - умовна точка переривання;
- - відключена умовна точка переривання;
- - лічильник програми;
- - лічильник програми і одиночна точка переривання;
- - лічильник програми і множинні точки переривання;
- - місце у вихідній програмі, звідки був зроблений виклик.

Для установки строкової точки переривання клацніть в рамці зліва від рядка або натисніть клавіші **Ctrl + F8**.

Для установки точки переривання одного з інших типів необхідно виконати наступні дії:

• виділіть елемент, для якого має бути задана точка переривання (наприклад, ім'я класу, ім'я методу або ім'я змінної);

• виконайте в меню **Run** команду **New Breakpoint** або натисніть клавіші **Ctrl + Shift + F8**;

• в діалоговому вікні команди **New Breakpoint** виберіть в полі **Breakpoint Type** тип переривання: **Class** (клас), **Exception** (виняток), **Method** (метод), **Thread** (потік) або **Field** (поле);

• встановіть в діалоговому вікні необхідні параметри в залежності від обраного типу переривання і натисніть кнопку **OK** (на рис. 13 наведено приклад установки переривання для змінної minValue при її зміні - Field Modification).

Для зміни параметрів існуючої точки переривання треба в меню **Window** вибрати команду **Debugging** і в підміню цієї команди вибрати команду **Breakpoints**. У відкритому внизу вікні **Breakpoints** (рис. 14) можна виконати наступні операції над точкою переривання:

• змінити параметри точки переривання за допомогою подвійного клацання миші по ній (буде відкрито вікно **Customize Breakpoint**, ідентичне вікну **New Breakpoint**);

• включити / вимкнути точку переривання за допомогою перемикача **Enable**;

• видалити точку переривання за допомогою команди **Delete** контекстного меню.

Включити / виключити строкову точку переривання можна, якщо викликати для неї контекстне меню і в команді **Breakpoint** вибрати опцію **Enabled**. Строкова точка переривання видаляється, якщо клацнути мишею по її значку.

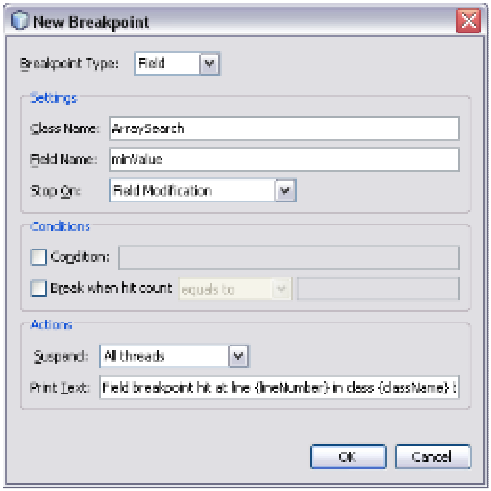

Рис. 13. Установка параметрів для точки переривання типу Variable

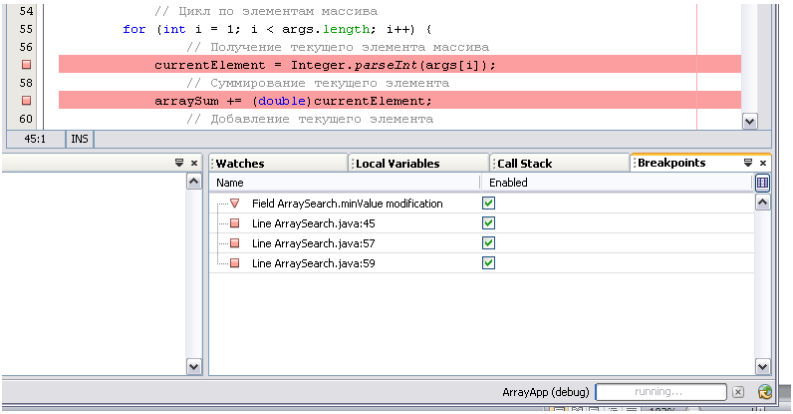

Рис. 14. Вікно Breakpoints

Для установки умовної точки переривання необхідно в поле Condition діалогового вікна **New Breakpoint** або **Customize Breakpoint** (рис. 13) задати умову з використанням синтаксису Java (наприклад, minValue <0, тобто переривання програми відбувається тільки в тому випадку, якщо значення minValue менше 0).

Покрокове виконання програми

При роботі в режимі налагодження рекомендується вивести в панелі інструментів панель **Debug** за допомогою включення опції **Debug** команди **Toolbars** меню **View**.

Запуск проекту в режимі налагодження виконується за допомогою команди **Debug Main Project** меню **Run** або при натисканні клавіші **F5**. Запустити проект в режимі налагодження можна також за допомогою команди **Debug Project** контекстного меню проекту.

Окремий файл проекту можна також запустити в режимі налагодження за допомогою команди **Debug File** контекстного меню файлу.

При запуску програми в режимі налагодження програма виконується до першої точки переривання або до кінця, якщо в програмі немає точок переривання.

Після досягнення першої точки переривання програму можна продовжувати виконувати по кроках, виконуючи одну з таких дій:

• продовжити виконання програми до наступної точки переривання або до кінця програми (якщо точок переривання більше немає);

• продовжити виконання програми до рядка, в якій знаходиться курсор;

• виконати наступну пропозицію програми і зупинити виконання.

Для продовження роботи програми до наступної точки переривання або до кінця програми треба або виконати команду **Continue** в меню **Run**, або натиснути на клавіші **Ctrl + F5**, або натиснути на кнопку  на панелі інструментів.

Щоб продовжити виконання програми до рядка, в якій знаходиться курсор, необхідно встановити курсор в необхідну рядок і або виконати команду **Run to Cursor** в меню **Run**, або натиснути на клавішу **F4**, або натиснути на кнопку  на панелі інструментів.

Виконання наступного речення програми і потім припинення виконання можна виконати за допомогою однієї з трьох команд меню **Run**:

• команда **Step Over** - виконує один рядок вихідного тексту, але якщо рядок містить виклик методу, виконує виклик даного методу як одне речення (цю команду можна виконати також за допомогою клавіші **F8** або кнопки  на панелі інструментів);

• команда **Step Into** - виконує один рядок вихідного тексту, але якщо рядок містить виклик методу, переходить в метод і призупиняє виконання перед першим виконуваним пропозицією методу (цю команду можна виконати також за допомогою клавіші **F7** або кнопки  на панелі інструментів);

• команда **Step Out** - виконує один рядок вихідного тексту, але якщо рядок міститься в викликаному методі, виконує всі пропозиції, що залишилися методу і повертає управління в метод, який викликав даний метод (цю команду можна виконати також за допомогою клавіші **Ctrl + F7** або кнопки  на панелі інструментів).

Інформації про хід виконання налагодження виводиться у вкладці **Debugger Console** вікна **Output** (рис. 15). Клік миші по посиланню вкладки

викликає перехід до рядка, що викликала переривання у вікні редактора програми.

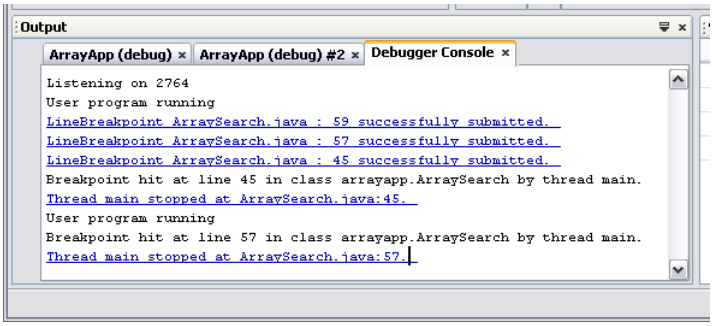

Рис. 15. Вкладка Debugger Console вікна Output

Припинити виконання програми в режимі налагодження можна або за допомогою команди **Finish Debugger Session** в меню **Run**, або натиснувши на клавіші **Shift + F5**, або натиснувши на кнопку  на панелі інструментів.

Стеження за змінними програми

При налагодженні поточні значення локальних змінних можна отримати у вікні **Local Variables** (рис. 16), яке виводиться на екран при включенні опції **Local Variables** в команді **Debugging** меню **Window**. Поточне значення локальної змінної можна при необхідності змінити, якщо натиснути кнопку праворуч від значення змінної.

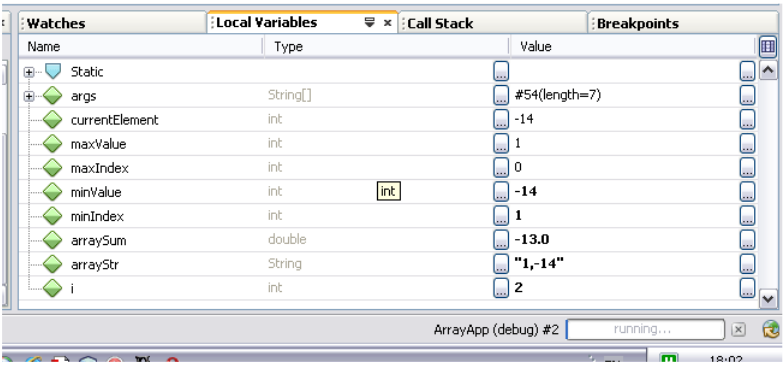

Рис. 16. Вікно Local Variables

Поточне значення змінної можна також отримати, якщо навести курсор миші на ім'я змінної у вікні редактора (рис. 17).

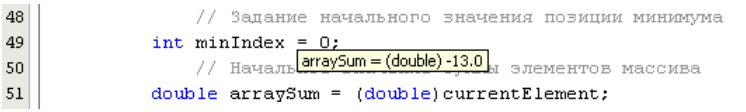

Рис. 17. Отримання поточного значення змінної arraySum

Щоб відстежити значення будь-якої змінної або виразу при виконанні програми треба використовуються так звані спостережувані змінні (**watches**).

Спостережувані змінні виводяться у вікні **Watches** (рис. 18), яке виводиться на екран при включенні опції **Local Variables** в команді **Debugging** меню **Window**.

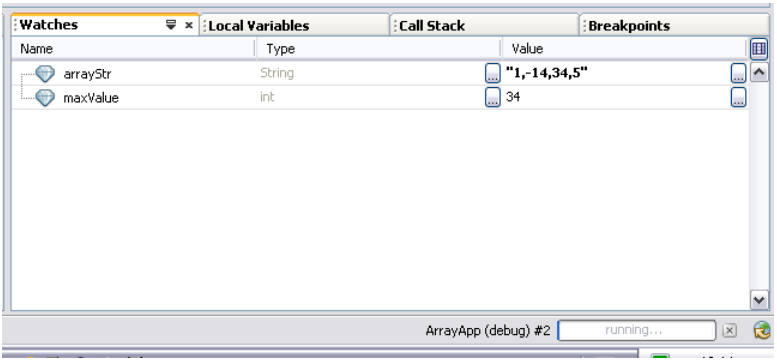

Рис. 18. Вікно Watches

Щоб створити спостережувану змінну або вираз, треба виділити їх у вікні редактора і або виконати для цієї змінної команду **New Watch** контекстного меню, або виконати команду меню **Run**, або натиснути кнопку на панелі інструментів, а потім натиснути кнопку **OK** в діалоговому вікні команди (рис. 19).

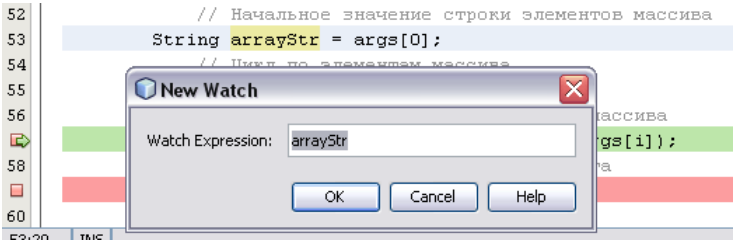

Рис. 19. Створення спостережуваної змінної

Значення спостережуваної змінної можна змінити тим же способом, що і значення локальної змінної.

Видалити спостережувану змінну можна за допомогою команди **Delete** контекстного меню змінної у вікні **Watches**.

3.1.3. Структура програми Java і простий введення-виведення

Якщо програма є додатком, вона повинна містити в файлі оголошення класу, ім'я якого збігається з ім'ям файлу, метод з ім'ям **main**. Цей метод повинен бути загальнодоступним, статичним і не повертає ніякого значення, а в якості параметрів цього методу повинен бути заданий масив рядків:

```
public static void main (String args [])
{
 тіло-методу
}
```

Метод main запускається першим при виконанні програми. У масиві строкових змінних **args** в програму передаються параметри, задані при запуску програми (перший параметр - в нульовому елементі масиву, другий в першому і т.д.). Опис масиву в параметрах методу **main** є обов'язковим, навіть якщо параметри не обробляються в програмі.

Параметри, що задаються в командному рядку, можна використовувати для введення даних в програму.

Для виведення результатів виконання програми та діагностичних повідомлень на дисплей в класі **System** пакету **java.lang** описані два потокових об'єкта класу **PrintStream**:

```
public static final PrintStream out
```

і

```
public static final PrintStream err
```

Оскільки це статичні об'єкти, їх можна використовувати в програмах, не створюючи власних потокових об'єктів, і в той же час їм доступні методи класу для виведення даних (зазвичай використовуються методи **print()** і **println()**).

При виклику методів **print()** і **println()** може бути заданий аргумент типу **String, char, char [], int, long, float, double, boolean** і **Object**, наприклад:

```
int i = 10;
System.out.println ( "i =" + i);
```

На дисплей буде виведений рядок: **i = 10**.

Єдина відмінність між методами **print()** і **println()** полягає в тому, що останній завжди додає символ перекладу рядка до всього, що виводить на дисплей, а перший залишається на тому самому рядку. Якщо потрібно перейти на новий рядок, можна викликати **println()** без параметрів.

Виконання програми закінчується, коли виконаний останній оператор програми або якщо в програмі сталася помилка (в цьому випадку об'єкт **System.err** виводить на дисплей діагностичні повідомлення).

Виконання програми можна також завершити за допомогою наступного методу класу **System**:

```
public static void exit (int status)
```

Параметр status задає значення коду завершення програми, наприклад:

```
boolean progError;
...
if (progError)
System.exit (5);
```

За замовчуванням програма закінчується з нульовим кодом завершення. Можна задати для параметра **status** ненульовий код завершення, який може бути проаналізований операційною системою або командним файлом.

3.2. Масиви в Java

Масивом в мовах програмування називається іменований набір змінних однакового типу. Кожен елемент масиву є змінною, яка однозначно визначається шляхом вказівки імені масиву і целочисленной позиції елемента в масиві - індексу елемента в масиві.

3.2.1. Опис масивів

Масиви в Java є гібридом об'єктів і примітивних типів. Вони схожі на об'єкти, але мають спеціальне значення для компілятора. Опис проводиться в три етапи.

На першому етапі виконується оголошення масиву (array declaration). Оголошення масиву в Java виконується з використанням одного з наступних форматів:

ім'я-типу [] ідентифікатор-масиву

або

ім'я-типу ідентифікатор-масиву []

де **ім'я-типу** - ім'я примітивного або посилального типу, а **ідентифікатор-масиву** - ім'я, що привласнюється масиву.

Якщо кілька масивів мають однаковий тип, їх також можна оголосити в списку, використовуючи перший формат запису.

На другому етапі, етапі визначення або створення масиву (array instantiation) вказується кількість елементів масиву, зване його довжиною, виділяється місце для масиву в оперативній пам'яті, змінна-посилання отримує адресу масиву. Ці дії виконуються наступного операцій:

new ім'я-типу [розмір-масиву]

де **розмір-масиву** - змінна або вираз будь-якого цілочисельного типу, за винятком типу **long**. Як видно з цієї операції, на відміну від визначення об'єкта класу, для масиву вказується не конструктор класу, а тип масиву і його розмір.

На третьому етапі проводиться ініціалізація масиву (array initialization). Елементи масивів числових типів отримують нульове значення відповідного типу, елементи булевских масивів - значення false, а елементи масивів посилальних типів - значення null.

Масиви можуть бути ініційовані під час створення. При цьому розмір-масиву в квадратних дужках не вказується, а після закриває квадратної дужки

в фігурних дужках задається список значень елементів масиву, розділених комами. Розмір масиву в даному разі відповідає кількості елементів в списку (в списку не повинно бути порожніх елементів).

Можна об'єднати перший і другий етапи в одному операторі, наприклад:

```
int [] myArray = new int [5];
```

Оскільки при ініціалізації масиву розмір масиву вказувати не треба, можна об'єднати перший і третій етап в одному операторі, наприклад:

```
int [] myArray = {0, 1, 2, 3, 4};
```

3.2.2. Доступ до елементів масиву

Елемент масиву, крім значення, характеризується своєю позицією в масиві - індексом. Індексом елемента є значення типу **byte**, **short**, **int** або **char**, укладену в квадратні дужки, наприклад **myArray [4].** Індекс елемента масиву змінюється від нуля до величини, на одиницю меншою розміру масиву.

Довжину масиву можна визначити за допомогою властивості масиву **length**.

4. Порядок виконання роботи

4.1. Відкомпілюйте і виконайте програму на мові Java в IDE NetBeans IDE 5.5.1 BlueJ Edition.

4.1.1. Запустіть IDE і відкрийте в ньому новий проект **HelloJava** з місцем розташування у своїй папці.

4.1.2. Вставте в метод **main** головного класу такі рядки:

```
 System.out.println ( "Привіт, Java!");
Date d = new Date ();
System.out.println ( "Дата:" + d.toString ());
```

4.1.3. Виправте помилку в рядку 27, викликавши контекстне меню для рядка помилки і виконавши команду **Fix Imports** для **java.util.Date** (в результаті перед визначенням класу буде додано пропозицію **import** для конструктора **Date**).

4.1.4. Збережіть файл **Main.java** і виконайте збірку проекту.

4.1.5. Запустіть проект на виконання.

4.1.6. Скопіювати вміст вкладки **Run HelloJava** вікна **Output** в текстовий файл для звіту (використовуючи виділення виведення програма і клавіші **Ctrl + C**).

4.2. Напишіть програму на мові Java по одному з наведених нижче варіантів.

Завдання початкових значень змінним виконується або при ініціалізації, або за допомогою оператора присвоювання, а елементи масиву задаються як аргументи при виклику програми. Якщо у варіанті вихідними є

два масиви, то елементи першого масиву задаються як аргументи при виклику програми, а елементи другого масиву задаються при його визначенні та ініціалізації. Тип масивів - int. Отримані результати виводяться на дисплей з пояснювальними повідомленнями.

Варіант 1

Сформувати і вивести на дисплей одновимірний масив b, в якому першими елементами є елементи вихідного одновимірного масиву a з негативними значеннями (зі збереженням порядку проходження), а потім елементи a з нульовими і позитивними значеннями.

Варіант 2

Визначити значення двох найбільших і різних за значенням елементів вихідного одновимірного масиву a і їх індекси (масив може містити елементи з рівними значеннями, тобто необхідно вивести значення і індекси елементів з максимальними значеннями і значення другого за величиною елемента, а також індекси всіх елементів , мають другу за величиною значення).

Варіант 3

Сформувати одновимірний масив b з вихідного одновимірного масиву a наступним чином: якщо значення будь-яких двох або більше елементів масиву a дорівнюють один одному, на місці всіх цих елементів в масиві b виводиться 1, в іншому випадку (якщо i-ий елемент не дорівнює ніякому іншого елементу) в масиві b виводиться 0.

Варіант 4

Сформувати одновимірний масив b з вихідного одновимірного масиву a шляхом циклічного зсуву елементів a на k позицій вправо. Значення k задається як перший аргумент при виклику програми, інші аргументи - елементи масиву.

Варіант 5

Визначити, чи є всі елементи вихідного одновимірного масиву a негативними величинами або вони все позитивні або серед елементів a є як позитивні, так і негативні величини і вивести відповідні повідомлення для кожного випадку.

Варіант 6

Визначити значення і індекси локальних мінімумів вихідного одновимірного масиву a (елемент масиву називається локальним мінімумом, якщо він строго менше своїх сусідів).

Варіант 7

Визначити абсолютне значення найменшої різниці між двома будь-якими значеннями елементів вихідного одновимірного масиву a.

Варіант 8

Визначити абсолютні значення найбільшою і найменшою різниці між середнім значенням і значеннями елементів вихідного одновимірного масиву a.

Варіант 9

Сформувати масив b з вихідного одновимірного масиву a за наступним алгоритмом: спочатку йдуть елементи масиву a з парними значеннями в порядку їх зростання, потім елементи з непарними значеннями в порядку їх зменшення. Для визначення кількості парних елементів використовуйте оператор взяття модуля "%".

Варіант 10

Сформувати масив b з вихідного одновимірного масиву a за наступним алгоритмом: b_i дорівнює кількості елементів із значенням, рівним ai, в масиві a.

Варіант 11

Визначити індекси і значення елементів вихідного одновимірного масиву a, величини яких лежать поза задається нижньої a_{min} і верхньої a_{max} кордонів ($a_i < a_{min}$ або $a_i > a_{max}$). Значення a_{min} і a_{max} задаються як перші два аргументи при виклику програми, інші аргументи - елементи масиву.

Варіант 12

Визначити, чи утворюють значення елементів вихідного одновимірного масиву a: строго зростаючу послідовність ($a_i < a_i + 1$), строго спадну послідовність ($a_i > a_i + 1$) або елементи масиву не впорядковані і вивести для кожного випадку відповідне повідомлення.

Варіант 13

Визначити, чи утворюють значення елементів вихідного одновимірного масиву a: арифметичну прогресію, тобто $a_i = a_i\text{-}1 + n$, де n - різниця прогресії і вивести відповідне повідомлення.

Варіант 14

Перевірте, чи є елементи масиву a безліччю (для цього серед елементів масиву не повинно бути двох елементів з однаковим значенням).

Варіант 15

Виведіть на дисплей значення тих елементи масивів a і b, які є і в тому, і в іншому масиві (передбачається, що і масив a і масив b є множинами, тобто кожен з них не містить елементів з однаковими значеннями).

Варіант 16

Виведіть на дисплей значення тих елементи масивів a і b, які є тільки в одному з масивів, і відсутні в іншому масиві (передбачається, що і масив a і масив b є множинами, тобто кожен з них не містить елементів з однаковими значеннями).

Варіант 17

Сформуйте з масиву a масив b за наступним алгоритмом: елемент масиву b дорівнює значенню різниці між максимальним значенням елементів масиву a і значенням даного елемента масиву a.

Варіант 18

Виведіть на дисплей розподіл значень елементів масиву a по інтервалах. Межі інтервалів задаються у вигляді масиву b, причому нульовий елемент визначає нижню межу першого інтервалу, а елементи з 1-го по n-ий -

верхні межі інтервалів. В результаті роботи програми на дисплей повинні бути виведені рядки «Інтервал nn - xx» і останній рядок «Поза інтервалів - xx».

Варіант 19

Виведіть на дисплей значення і індекси тільки тих елементів масиву a, значення яких не дорівнюють значенням інших елементів, тобто унікальних елементів масиву.

Варіант 20

Визначте (у відсотках від загальної кількості елементів), скільки елементів в масиві a мають значення менше, ніж середнє значення, скільки елементів - значення, рівне середньому значенню і скільки елементів мають значення, більше, ніж середнє значення.

Варіант 21

Перевірте, чи не є значення i-их елементів масиву a лінійної комбінацією i-их значень елементів масиву b, тобто ai = k * bi + c, де k і c - константи (значення k і c можна визначити з значень двох перших елементів a і b як два рівняння з двома невідомими).

Варіант 22

Сформувати масив b, елементами якого є значення індексів елементів вихідного одновимірного масиву a в порядку убування значень елементів.

Варіант 23

Визначити індекси і значення рівних елементів (якщо вони є) вихідного одновимірного масиву ().

Варіант 24

Визначте номер дня в році по заданому номеру дня в місяці і номеру місяця (вводяться як аргументи при виклику програми). Ознака, чи є рік високосним, задається як булевська змінна. Вказівка: кількість днів до початку цього місяця (НЕ високосний рік): січень 0, лютий - 31, березень - 59, квітень - 90, травень -120, червень - 151, липень - 181, серпень - 212, вересень - 243, жовтень - 273, листопад - 314, грудень - 334 задати у вигляді масиву. У високосному році, починаючи з березня, до кількості днів додається 1.

Варіант 25

Визначте номер дня у місяці та номер місяця року по заданому номеру дня в році (вводиться як аргумент при виклику програми). Ознака, чи є рік високосним, задається як булевська змінна. Вказівка: кількість днів до початку цього місяця (НЕ високосний рік): січень - 0, лютий - 31, березень - 59, квітень - 90, травень -120, червень - 151, липень - 181, серпень - 212, вересень - 243, жовтень - 273, листопад - 314, грудень - 334 задати у вигляді масиву. У високосному році, починаючи з березня, до кількості днів додається 1.

Варіант 26

Сформувати масив b, елементами якого є елементи вихідного одновимірного масиву a, розташовані в зворотному порядку.

Варіант 27

Знайти і замінити в вихідному одновимірному масиві a все значення $a_i = k_f$ на k_c. Значення k_f і k_c вводяться як перші два аргументи при виклику програми (інші аргументи - елементи масиву).

Варіант 28

Визначити кількість рівних елементів ($a_i = b_i$) і їх індекси для двох вихідних одновимірних масивів a і b.

Варіант 29

Сформувати масив b з вихідного одновимірного масиву a наступним чином: якщо $a_{min} < a_i < a_{max}$, то $b_i = a_i$; якщо $a_i \le a_{min}$, то $b_i = a_{min}$; якщо ai ≥ amax, то $b_i = a_{max}$ (a_{max} і a_{min} - максимальне і мінімальне значення елементів в масиві).

Варіант 30

Сформувати масив b з масиву a наступним чином: масив b складається з тих елементів масиву a, які повторюються в масиві (по одному значенню для однакових елементів), наприклад, для масиву a: 3 7 4 3 8 7 5, масив b матиме вигляд 3 7.

5. Приклад виконання завдання 4.2

Пошук максимального, мінімального і середнього значення в масиві. Для максимального і мінімального значень задаються також їх індекси в масиві. Елементи масиву - цілі числа, задаються як аргументи при запуску програми.

Текст програми:

```
package arraysort;
public class ArraySearch {  // Конструктор ArraySearch без параметрів
public ArraySearch () {
}
public static void main (String [] args) {
if (args.length == 0) {
// Якщо аргументи при виклику програми не задані
System.out.println ( "Не задані елементи масиву");
// Виведення повідомлення про помилку
return; // Вихід з функції
}
int currentElement = Integer.parseInt (args [0]);
// Визначення числового значення першого елемента масиву
int maxValue = currentElement;
// Завдання початкового значення максимуму
int maxIndex = 0;  // Завдання початкового значення позиції максимуму
int minValue = currentElement;
// Завдання початкового значення мінімуму
int minIndex = 0; // Завдання початкового значення позиції мінімуму
double arraySum = (double) currentElement;
// Початкове значення суми елементів масиву
```

```
String arrayStr = args [0];
// Початкове значення рядка елементів масиву
for (int i = 1; i <args.length; i ++) {
// Цикл за елементами масиву
currentElement = Integer.parseInt (args [i]);
// Отримання поточного елемента масиву
arraySum + = (double) currentElement;
// Підсумовування поточного елемента
arrayStr + = "," + args [i];
// Додавання поточного елемента масиву в рядок
if (currentElement> maxValue) {
// Якщо поточний елемент більше максимуму
maxValue = currentElement;
// Присвоєння максимуму значення поточного елемента
maxIndex = i; // Присвоєння нового значення позиції максимуму
}
if (currentElement <minValue) {
// Якщо поточний елемент менше мінімуму
minValue = currentElement;
 // Присвоєння мінімуму значення поточного елемента
minIndex = i; // Присвоєння нового значення позиції мінімуму
}
}
System.out.println ( "Елементи масиву:" + arrayStr);
// Вивід елементів масиву
System.out.println ( "Максимум:" + maxValue + "Індекс:" +
maxIndex); // Вивід максимуму і позиції максимуму
System.out.println ( "Мінімум:" + minValue + "Індекс:" +
minIndex); // Вивід мінімуму і позиції мінімуму
System.out.println ( "Середнє значення:" +(ArraySum /
args.length)); // Вивід середнього значення
}
}
```

Приклад виведення:

Елементи масиву: 1, -14,34,5,7,42,12

Максимум: 42 Індекс: 5

Мінімум: -14 Індекс: 1

Середнє значення: 12.428571428571429

6. Зміст звіту

У звіті повинні бути представлені:

• текст і висновок програми HelloJava;

• текст і висновок програми, а також скріншот вікна Project Properties (рис. 12) з аргументами виклику для свого варіанту.

7. Питання для самоконтролю

1. Як записується програма на мові Java, і які типи елементів програми визначені в мові Java? Коротко охарактеризуйте кожен тип.

2. За якими правилами формуються ідентифікатори в мові Java?

3. Як оголошується клас в мові Java?

4. Для яких цілей використовуються пакети в мові Java? Які компоненти входять до складу пакетів?

5. Як в програмі на мові Java включається необхідний клас або класи будь-якого пакета? Як зробити змінні і методи класу доступними у всіх пакетах?

6. Які категорії даних визначені в мові Java?

7. Які примітивні типи даних визначені в мові Java? Коротко охарактеризуйте кожен тип даних.

8. Як оголошуються змінні в мові Java? Як визначається область дії і час життя змінної в програмі на мові Java?

9. Які модифікатори і як змінюють область дії і час життя змінної в програмі на мові Java?

10. Які типи цілих змінних існують в мові Java і як визначаються значення цілих змінних?

11. Як визначаються символьні і булевские змінні в мові Java?

12. Які типи дійсних змінних існують в мові Java і як визначаються значення речових змінних?

13. Як визначається і як використовується операція присвоювання в мові Java?

14. Які унарні операції визначені в мові Java? Дайте коротку характеристику кожної операції.

15. Які арифметичні бінарні операції визначені в мові Java? Дайте коротку характеристику кожної операції.

16. Які побітові бінарні операції визначені в мові Java? Дайте коротку характеристику кожної операції.

17. Як визначається складова операція в мові Java, і для яких операцій вона застосовується?

18. Які операції порівняння визначені в мові Java? Дайте коротку характеристику кожної операції.

19. Які булевские операції визначені в мові Java? Дайте коротку характеристику кожної операції. Як визначаються укорочені булевские операції і чим вони відрізняються від звичайних булевских операцій?

20. Як записується і як функціонує умовна операція в Java?

21. Як визначаються пріоритети операцій в Java? Як можна змінити порядок виконання операцій?

22. Як виконуються розширюють перетворення типів в мові Java?

23. Як виконуються звужують перетворення типів в мові Java?

24. Які функції виконують модифікатори private і native?

25. Як описуються і не започатковано масиви в мові Java?

26. Як виконується доступ до елементів масиву в мові Java?

27. Як визначаються багатовимірні масиви в мові Java?

28. Які види операторів управління визначені в мові Java і де використовується кожен вид?

29. Як визначається і як функціонує умовний оператор в мові Java?

30. Які оператори циклу визначені в мові Java? Як описується і як функціонує кожен з цих операторів?

31. Як і де в Java використовується оператор кома?

32. Які оператори переходу визначені в мові Java і як вони функціонує? Для яких цілей в операторах переходу використовуються мітки?

33. Як визначається і як функціонує оператор вибору в мові Java? Для чого в операторі вибору використовується оператор break?

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
для виконання лабороторних робіт на тему
«Технологія програмування на мові Java. Робота з масивами в Java»
із дисциплін
«Програмування мобільних комп'ютерних систем»,
«Прикладне програмування в телекомунікаційних системах»,
«Програмування мобільних пристроїв телекомунікацій»
для студентів спеціальностей
6.171 "Електроніка",
6.172 "Телекомунікації та радіотехніка"
денної форми навчання

Відповідальний за випуск А. С. Опанасюк
Редактор Н. М. Мажура
Комп’ютерне верстання О. Є. Горячева

Формат 60х84/16. Ум. друк. арк. 1,16. Обл.-вид. арк. 1,09.

Видавець і виготовлювач
Сумський державний університет,
вул. Римського-Корсакова, 2, м. Суми, 40007
Свідоцтво суб’єкта видавничої справи ДК № 3062 від 17.12.2007.